UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/805,963	03/22/2004	Kent F. Hayes JR.	RSW920030236US1	2602

23550 7590 04/17/2007

HOFFMAN WARNICK & D'ALESSANDRO, LLC
75 STATE STREET
14TH FLOOR
ALBANY, NY 12207

EXAMINER	
WANG, BEN C	
ART UNIT	**PAPER NUMBER**
2192	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	04/17/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No.	Applicant(s)	
	10/805,963	HAYES, KENT F.	
	Examiner	Art Unit	
	Ben C. Wang	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *22 March 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-40* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-40* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *03/22/2004*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-40 are pending in this application and presented for examination.

Specification Objections

2. The specification is objected to because the following informalities:

- "JAVA", cited in [0018], Line 5, is a registered trademark
- "OSGi bundle 16A will be registered with server 20", cited in [0021], Line 13, should be corrected as "OSGi bundle 16A will be registered with server 12"
- "In any event, deployment system 58 will thereafter deploy te native code", cited in [0035], Line 7, should be corrected as "In any event, deployment system 58 will thereafter deploy the native code"
- "the native application has prerequisites, the native application is packages within an OSGi" cited in [0036], Line 6, should be corrected as "the native application has prerequisites, the native application is packaged within an OSGi"

Appropriate correction is required.

Claim Objections

3. Claim 29 is objected to because the following informalities:

- "The system of of claim 19, wherein the dependency", claim 29, line 1, should be corrected as "The system of claim 19, wherein the dependency"

Appropriate correction is required.

Claim Rejections – 35 USC § 103(a)

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1-9, 11-26, 28-29, and 31-39 are rejected under 35 U.S.C. 103(a) as being unpatentable over Klicnik et al. (Pub. No. US 2002/0184226 A1) (hereinafter 'Klicnik') in view of Liang et al. (*Bundle Dependency in Open Services Gateway Initiative Framework Initialization, 2002, IEEE*) (hereinafter 'Liang')

6. **As to claim 1**, Klicnik discloses a computer-implemented method for resolving prerequisites for native applications comprising: packaging a native application for a client device and corresponding dependency information within a first OSGi bundle on a server ([0010], Lines 10-16 – A bundle's manifest file identifies the bundle's contents and also the packages and services which are imported and exported by that bundle), wherein the corresponding dependency information specifies at least one prerequisite on which the native application depends for proper operation on the client device (Fig. 2; [0031], Lines 9-24; Fig. 3 – all prerequisites items; [0032]); polling the client device to determine if the client device has the at least one other prerequisite (Figs. 4A-4C; Fig. 5; [0027]; [0043], Lines 1-3); obtaining the at least one prerequisite if the client device

does not have the at least one prerequisite ([0035] – a plug-in's specification may include a filtering expression or parameter, referred to herein as the "expose" parameter, to explicitly detail the classes a class loader will load on behalf of class loaders from other plug-ins); and loading the at least one prerequisite and the native application on the client device (Abstract, Lines 1-4 – dynamically loading components which have prerequisite relationships more complex than the simple single inheritance chains which are supported with prior art dynamic class loading techniques; [0015] – dynamically loading components which allows explicitly specifying one or more prerequisite components; [0018], Lines 5-13 – providing a specification of zero or more prerequisite components for components to be loaded).

Although Klicnik discloses OSGI bundles ([0010]), but does not explicitly disclose resolving prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework.

However, in an analogous art of Bundle Dependency in Open Services Gateway Initiative Framework Initialization, Liang discloses resolving prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework (Abstract, Lines 1-11; Sec. 1 of Introduction, 1st Para., Lines 1-6; 2nd Para., Lines 1-12; Sec. of II Bundle Dependency During Framework Initialization, 1st Para., Lines 1-3; Fig. 3 – bundle dependency relationship; Sec. of IV. Conclusions and Discussions, 1st Para., Lines 1-2 – some of the solutions provided here are constructed form the OSGi server side).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Liang into the Klicnik's system

to further resolve prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework.

The motivation is that it would enhance the Klicnik's system by taking, advancing and/or incorporating Liang's system which provides the framework can look up all events of those bundles to manage the bundle dependency automatically as once suggested by Liang (i.e., sec. of e) A New OSGi Component Model, 1st Para., Lines 13-16).

7. **As to claim 11**, Klicnik discloses a computer-implemented method for resolving prerequisites for native applications, comprising: packaging a native application for a client device and corresponding dependency information within a first bundle on a server ([0010], Lines 10-16 – A bundle's manifest file identifies the bundle's contents and also the packages and services which are imported and exported by that bundle), wherein the dependency information specifies at least one prerequisite on which the native application depends for proper operation on the client device (Fig. 2; [0031], Lines 9-24; Fig. 3 – all prerequisites items; [0032]); polling the client device to determine if the client device has the at least one other prerequisite (Figs. 4A-4C; Fig. 5; [0027]; [0043], Lines 1-3); obtaining the at least one prerequisite if the client device does not have the at least one prerequisite, wherein the at least one prerequisite is packaged within a second bundle that is accessible to the server ([0035] – a plug-in's specification may include a filtering expression or parameter, referred to herein as the "expose" parameter, to explicitly detail the classes a class loader will load on behalf of class

loaders from other plug-ins); and installing the first bundle and the second bundle within an environment of the client device (Abstract, Lines 1-4 – dynamically loading components which have prerequisite relationships more complex than the simple single inheritance chains which are supported with prior art dynamic class loading techniques; [0015] – dynamically loading components which allows explicitly specifying one or more prerequisite components; [0018], Lines 5-13 – providing a specification of zero or more prerequisite components for components to be loaded).

Although Klicnik discloses OSGI bundles ([0010]), but does not explicitly disclose resolving prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework.

However, in an analogous art of Bundle Dependency in Open Services Gateway Initiative Framework Initialization, Liang discloses resolving prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework (Abstract, Lines 1-11; Sec. 1 of Introduction, 1st Para., Lines 1-6; 2nd Para., Lines 1-12; Sec. of II Bundle Dependency During Framework Initialization, 1st Para., Lines 1-3; Fig. 3 – bundle dependency relationship; Sec. of IV. Conclusions and Discussions, 1st Para., Lines 1-2 – some of the solutions provided here are constructed form the OSGi server side).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Liang into the Klicnik's system to further resolve prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework.

The motivation is that it would enhance the Klicnik's system by taking, advancing and/or incorporating Liang's system which provides the framework can look up all events of those bundles to manage the bundle dependency automatically as once suggested by Liang (i.e., sec. of e) A New OSGi Component Model, 1st Para., Lines 13-16).

8. **As to claim 19**, Klicnik discloses a computerized system for resolving prerequisites for native applications, comprising: a packaging system for packaging a native application for a client device and corresponding dependency information within a first bundle on a server ([0010], Lines 10-16 – A bundle's manifest file identifies the bundle's contents and also the packages and services which are imported and exported by that bundle), wherein the dependency information specifies at least one prerequisite on which the native application depends for proper operation on the client device (Fig. 2; [0031], Lines 9-24; Fig. 3 – all prerequisites items; [0032]); a communication system for polling the client device to determine if the client device has the at least one other prerequisite (Figs. 4A-4C; Fig. 5; [0027]; [0043], Lines 1-3); a resolution system for obtaining the at least one prerequisite if the client device does not have the at least one prerequisite, wherein the at least one prerequisite is packaged within a second bundle that is accessible to the server ([0035] – a plug-in's specification may include a filtering expression or parameter, referred to herein as the "expose" parameter, to explicitly detail the classes a class loader will load on behalf of class loaders from other plug-ins); and a bundle loading system for loading the first bundle and the second bundle on the

client device (Abstract, Lines 1-4 – dynamically loading components which have prerequisite relationships more complex than the simple single inheritance chains which are supported with prior art dynamic class loading techniques; [0015] – dynamically loading components which allows explicitly specifying one or more prerequisite components; [0018], Lines 5-13 – providing a specification of zero or more prerequisite components for components to be loaded).

Although Klicnik discloses OSGI bundles ([0010]), but does not explicitly disclose resolving prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework.

However, in an analogous art of Bundle Dependency in Open Services Gateway Initiative Framework Initialization, Liang discloses resolving prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework (Abstract, Lines 1-11; Sec. 1 of Introduction, 1st Para., Lines 1-6; 2nd Para., Lines 1-12; Sec. of II Bundle Dependency During Framework Initialization, 1st Para., Lines 1-3; Fig. 3 – bundle dependency relationship; Sec. of IV. Conclusions and Discussions, 1st Para., Lines 1-2 – some of the solutions provided here are constructed form the OSGi server side).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Liang into the Klicnik's system to further resolve prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework.

The motivation is that it would enhance the Klicnik's system by taking, advancing and/or incorporating Liang's system which provides the framework can look up all

events of those bundles to manage the bundle dependency automatically as once suggested by Liang (i.e., sec. of e) A New OSGi Component Model, 1st Para., Lines 13-16).

9. **As to claim 31**, Klicnik discloses a program product stored on a recordable medium for resolving prerequisites for native applications, which when executed, comprises: program code for packaging a native application for a client device and corresponding dependency information within a first bundle on a server ([0010], Lines 10-16 – A bundle's manifest file identifies the bundle's contents and also the packages and services which are imported and exported by that bundle), wherein the dependency information specifies at least one prerequisite on which the native application depends for proper operation on the client device (Fig. 2; [0031], Lines 9-24; Fig. 3 – all prerequisites items; [0032]); program code for polling the client device to determine if the client device has the at least one other prerequisite (Figs. 4A-4C; Fig. 5; [0027]; [0043], Lines 1-3); program code for obtaining the at least one prerequisite if the client device does not have the at least one prerequisite, wherein the at least one prerequisite is packaged within a second bundle that is accessible to the server ([0035] – a plug-in's specification may include a filtering expression or parameter, referred to herein as the "expose" parameter, to explicitly detail the classes a class loader will load on behalf of class loaders from other plug-ins); and program code for loading the first bundle and the second bundle on the client device (Abstract, Lines 1-4 – dynamically loading components which have prerequisite relationships more complex than the simple single

inheritance chains which are supported with prior art dynamic class loading techniques; [0015] – dynamically loading components which allows explicitly specifying one or more prerequisite components; [0018], Lines 5-13 – providing a specification of zero or more prerequisite components for components to be loaded).

Although Klicnik discloses OSGI bundles ([0010]), but does not explicitly disclose resolving prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework.

However, in an analogous art of Bundle Dependency in Open Services Gateway Initiative Framework Initialization, Liang discloses resolving prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework (Abstract; Lines 1-11; Sec. 1 of Introduction, 1st Para., Lines 1-6; 2nd Para., Lines 1-12; Sec. of II Bundle Dependency During Framework Initialization, 1st Para., Lines 1-3; Fig. 3 – bundle dependency relationship; Sec. of IV. Conclusions and Discussions, 1st Para., Lines 1-2 – some of the solutions provided here are constructed form the OSGi server side).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Liang into the Klicnik's system to further resolve prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework.

The motivation is that it would enhance the Klicnik's system by taking, advancing and/or incorporating Liang's system which provides the framework can look up all events of those bundles to manage the bundle dependency automatically as once

suggested by Liang (i.e., sec. of e) A New OSGi Component Model, 1st Para., Lines 13-16).

10. **As to claim 2** (incorporating the rejection in claim 1), Liang discloses the method, further comprising registering the packaged native application and first OSGi bundle after the packaging step, wherein the registering step comprises storing the corresponding dependency information (Sec. 1, 1st Para., Lines 12-14 – open services include service discovery, service registration, service deployment, service processing, and service security, 2nd Para., Lines 17-19 – from a shared service registry; Sec. of d) By Using ServicEvent and Event Handling Mechanism of OSGi, 1st Para., Lines 2-12).

11. **As to claims 3** (incorporating the rejection in claim 1) **and 13** (incorporating the rejection in claim 11), Klicnik discloses the method, wherein the polling step comprises: identifying the at least one prerequisite to the client device (Figs. 4A-4C; Fig. 5; [0027]; [0043], Lines 1-3); and Liang discloses receiving a response from the client device, wherein the response indicates whether the client device has the at least one prerequisite (Sec. of b) Modify the Bundle Management Strategy of the Framework, Lines 1-4).

12. **As to claims 4** (incorporating the rejection in claim 1) **and** (incorporating the rejection in claim 11) **14**, Klicnik discloses the method, further comprising: determining the at least one prerequisite, prior to the packaging step; and generating the

corresponding dependency information based on the at least one prerequisite (Figs. 4A-4C; Fig. 5; [0027]; [0043], Lines 1-3).

13. **As to claims 5** (incorporating the rejection in claim 1) **and 15** (incorporating the rejection in claim 11), Klicnik discloses the method, wherein the at least one prerequisite comprises another native application ([0018], Lines 5-13).

14. **As to claim 6** (incorporating the rejection in claim 1), Liang discloses the method, wherein the at least one prerequisite is packaged with corresponding dependency information within a second OSGi bundle, and wherein the obtaining step comprises obtaining the second OSGi bundle (Fig. 3 – Bundle Dependency Relationship; Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 3rd Para.).

15. **As to claim 7** (incorporating the rejection in claim 6), Liang discloses the method, wherein loading step comprises: installing the first OSGi bundle and the second OSGi bundle within an OSGi environment of the client device (Sec. 1 of Introduction, 2nd Para., Lines 13—20; Sec. of Bundle Dependency During Framework Initialization, 3rd Para.); deploying the first OSGi bundle and the second OSGi bundle within a native environment of the client device (Sec. of Introduction, 2nd Para., Lines 1-12); and removing the native application from within the first OSGi bundle and the at least one prerequisite from within the second OSGi bundle (Sec. of c) Let The Third

Party Bundle To Manage the Service Dependency In a Centralized Control Way, 2nd Para., Lines 8-10).

16. **As to claims 8** (incorporating the rejection in claim 1) **and 18** (incorporating the rejection in claim 11), Klicnik discloses the method, wherein the method is performed recursively (Fig. 4A, step 430 – recursively look in class loader's parent; [0039], Lines 1-4; Fig. 4B, step 485 – recursively look in all prerequisite class loaders who export; [0041], Lines 10-11).

17. **As to claims 9** (incorporating the rejection in claim 1) **and 39** (incorporating the rejection in claim 31), Liang discloses the method and the program product, wherein the dependency information is expressed as a package import statement (Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 3rd Para., Lines 5-11 – import-package field in its manifest file).

18. **As to claim 12** (incorporating the rejection in claim 11), Liang discloses the method, wherein the first OSGi bundle and the second OSGi bundle are registered on the server after being packaged with the first native application and the at least one prerequisite (Sec. 1, 1st Para., Lines 12-14 – open services include service discovery, service registration, service deployment, service processing, and service security, 2nd Para., Lines 17-19 – from a shared service registry; Sec. of d) By Using ServicEvent and Event Handling Mechanism of OSGi, 1st Para., Lines 2-12).

19. **As to claim 16** (incorporating the rejection in claim 11), Liang discloses the method, further comprising: deploying the first OSGi bundle and the second OSGi bundle within a native environment of the client device (Sec. of Introduction, 2nd Para., Lines 1-12); and removing the native application from within the first OSGi bundle and the at least one prerequisite from within the second OSGi bundle (Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 2nd Para., Lines 8-10).

20. **As to claim 17** (incorporating the rejection in claim 11), Liang discloses the method, wherein the at least one prerequisite is packaged with corresponding dependency information within the second OSGi bundle (Fig. 3 – Bundle Dependency Relationship; Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 3rd Para.).

21. **As to claim 20** (incorporating the rejection in claim 19), Liang discloses the system, wherein packaging system further registers the first OSGi bundle after being packaged with the first native application (Sec. 1, 1st Para., Lines 12-14 – open services include service discovery, service registration, service deployment, service processing, and service security, 2nd Para., Lines 17-19 – from a shared service registry; Sec. of d) By Using ServicEvent and Event Handling Mechanism of OSGi, 1st Para., Lines 2-12).

22. **As to claim 21** (incorporating the rejection in claim 19), Klicnik discloses the system, wherein the communication system identifies the at least one prerequisite to the client device and receives a response from the client device that indicates whether the client device has the at least one prerequisite (Figs. 4A-4C; Fig. 5; [0027]; [0043], Lines 1-3).

23. **As to claim 22** (incorporating the rejection in claim 19), Klicnik discloses the system, further comprising: a prerequisite identification system for determining the at least one prerequisite; and an information generation system for generating the dependency information based on the at least one prerequisite (Figs. 4A-4C; Fig. 5; [0027]; [0043], Lines 1-3).

24. **As to claim 23** (incorporating the rejection in claim 19), Klicnik discloses the system, wherein the at least one prerequisite comprises another native application ([0018], Lines 5-13).

25. **As to claim 24** (incorporating the rejection in claim 19), Liang discloses the system, wherein bundle loading system comprises: an export system for installing the first OSGi bundle and the second OSGi bundle within the OSGi environment of the client device (Sec. 1 of Introduction, 2nd Para., Lines 13—20; Sec. of Bundle Dependency During Framework Initialization, 3rd Para.); a deployment system for deploying the first OSGi bundle and the second OSGi bundle within a native

environment of the client device (Sec. of Introduction, 2nd Para., Lines 1-12); and a removal system for removing the native application from within the first OSGi bundle and the at least one prerequisite from within the second OSGi bundle (Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 2nd Para., Lines 8-10).

26. **As to claim 25** (incorporating the rejection in claim 19), Liang discloses the system, wherein the at least one prerequisite is packaged with corresponding dependency information within the second OSGi bundle (Fig. 3 – Bundle Dependency Relationship; Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 3rd Para.).

27. **As to claim 26** (incorporating the rejection in claim 19), Klicnik discloses the system, wherein the client device includes: an analysis system for determining whether the client device has the at least one prerequisite; and a response system for generating and sending a response to the server (Figs. 4A-4C; Fig. 5; [0027]; [0043], Lines 1-3).

28. **As to claim 28** (incorporating the rejection in claim 27), Klicnik discloses the system, wherein the at least one prerequisite comprises another native application ([0018], Lines 5-13).

29. **As to claim 29** (incorporating the rejection in claim 19), Liang discloses the system, wherein the dependency information is expressed as a package import statement (Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 3rd Para., Lines 5-11 – import-package field in its manifest file).

30. **As to claim 32** (incorporating the rejection in claim 31), Liang discloses the program product, wherein program code for packaging further registers the first OSGi bundle after being packaged with the first native application (Sec. 1, 1st Para., Lines 12-14 – open services include service discovery, service registration, service deployment, service processing, and service security, 2nd Para., Lines 17-19 – from a shared service registry; Sec. of d) By Using ServicEvent and Event Handling Mechanism of OSGi, 1st Para., Lines 2-12).

31. **As to claim 33** (incorporating the rejection in claim 31), Klicnik discloses the program product, wherein the program code for polling identifies the at least one prerequisite to the client device and receives a response from the client device that indicates whether the client device has the at least one prerequisite (Figs. 4A-4C; Fig. 5; [0027]; [0043], Lines 1-3).

32. **As to claim 34** (incorporating the rejection in claim 31), Klicnik discloses the program product, further comprising: program code for determining the at least one

prerequisite; and program code for generating the dependency information based on the at least one prerequisite (Figs. 4A-4C; Fig. 5; [0027]; [0043], Lines 1-3).

33. **As to claim 35** (incorporating the rejection in claim 31), Klicnik discloses the program product, wherein the at least one prerequisite comprises another native application ([0018], Lines 5-13).

34. **As to claim 36** (incorporating the rejection in claim 31), Liang discloses the program product, further comprising: program code for installing the first OSGi bundle and the second OSGi bundle within an OSGi environment of the client device (Sec. 1 of Introduction, 2nd Para., Lines 13—20; Sec. of Bundle Dependency During Framework Initialization, 3rd Para.); program code for deploying the first OSGi bundle and the second OSGi bundle within a native environment of the client device (Sec. of Introduction, 2nd Para., Lines 1-12); and a removal system for removing the native application from within the first OSGi bundle and the at least one prerequisite from within the second OSGi bundle (Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 2nd Para., Lines 8-10).

35. **As to claim 37** (incorporating the rejection in claim 31), Klicnik discloses the program product, wherein the at least one prerequisite is packaged with corresponding dependency information within the second OSGi bundle ([0018], Lines 5-13).

36. **As to claim 38** (incorporating the rejection in claim 31), Klicnik discloses the program product, wherein the client device includes: program code for determining whether the client device has the at least one prerequisite; and program code for generating and sending a response to the server (Figs. 4A-4C; Fig. 5; [0027]; [0043], Lines 1-3).

37. Claims 10, 27, 30, and 40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Klicnik in view of Liang and in further view of Hall et al., (*Component Deployment on OSGi: The Gravity Case, January 29, 2003, Fractal Workshop – LSR-Adele*) (hereinafter 'Hall')

38. **As to claims 10** (incorporating the rejection in claim 1), **30** (incorporating the rejection in claim 19), and **40** (incorporating the rejection in claim 31), although Klicnik discloses OSGI bundles ([0010]) and Liang discloses bundle dependency during framework initialization (Sec. of II), but both do not explicitly disclose the method, the system, and the program product, wherein a name and version of the native application is represented in a name and version of the OSGi bundle.

However, in an analogous art of component deployment on OSGi: the gravity case, Hall discloses the method, the system, and the program product, wherein a name and version of the native application is represented in a name and version of the OSGi bundle (i.e., Slide 7 – Bundle Manifest Example - Import-Package, specification-version).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Hall into the Klicnik-Liang's system to further provide the method, the system, and the program product, wherein a name and version of the native application is represented in a name and version of the OSGi bundle.

The motivation is that it would enhance the Klicnik-Liang's system by taking, advancing and/or incorporating Hall's system which provides the framework of a factory service concept built on top of OSGi and further standardizes OSGi component creation as once suggested by Hall (i.e., Slides 30-31, 35-41 – Extended OSGi Component Model for Gravity).

39. **As to claim 27** (incorporating the rejection in claim 19), although Klicnik discloses OSGI bundles ([0010]) and Liang discloses bundle dependency during framework initialization (Sec. of II), but both do not explicitly disclose the system, wherein the dependency information specifies an identity and a version of the at least one prerequisite required by the native application.

However, in an analogous art of component deployment on OSGi: the gravity case, Hall discloses the system, wherein the dependency information specifies an identity and a version of the at least one prerequisite required by the native application (i.e., Slide 7 – Bundle Manifest Example - Import-Package, specification-version).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Hall into the Klicnik-Liang's

system to further provide the system, wherein the dependency information specifies an identity and a version of the at least one prerequisite required by the native application.

The motivation is that it would enhance the Klicnik-Liang's system by taking, advancing and/or incorporating Hall's system which provides the framework of a factory service concept built on top of OSGi and further standardizes OSGi component creation as once suggested by Hall (i.e., Slides 30-31, 35-41 – Extended OSGi Component Model for Gravity).

Conclusion

40. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- Jensen et al., *Protocol Method for Provisioning Services* (Pub. No. US 2004/0260774 A1)
- Mitchell et al., *Method and System for Dynamically Reconfiguring Pervasive Device Communication Channels* (Pub. No. US 2004/0117494 A1)
- Thorsteinsson et al., *Method and System for Controlling and Coordinating Devices and Appliances, such as from a Central Portal and via a Wide-Area Communications Networks* (Pub. No. US 2003/0105854 A1)
- Klicnik et al., *Independent Class Loader for Dynamic Class Loading* (Pat. No. US 6,748,396 B2)
- Rigori et al., *Extendable Provisioning Mechanism for a Service Gateway* (Pat. No. US 7,191,232 B2)
- Marples et al., *The Open Services Gateway Initiative: An Introductory Overview, 2001, IEEE*
- Kawamura et al., *Standardization Activity of OSGi (Open Services Gateway Initiative), January, 2004, NTT Technical Review*

41. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-

1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

BCW

TUAN DAM
SUPERVISORY PATENT EXAMINER

April 10, 2007